Scalability of Security in Distributed Object Systems: Panel Session

Participants: Bret Hartman, Odyssey Research Associates;
Dan Nessett, Sun Microsystems (Chair);
Nicholas Yialelis, Imperial College London

Abstract

This panel session addresses the problem of scalability in distributed object systems. It first describes the scaling problem and then uses several examples as discussion points for the participants.

Distributed object systems share many properties with traditional language based object systems (e.g., programs written in C++). For example, in both types of systems an object is accessed through an interface that hides its implementation. This allows a programmer to modify the object without forcing a concomitant modification of the program that uses the object.

However, there are also significant differences between these classes of system. Distributed object systems run on a platform consisting of more than one machine, while language based object systems run within a single process. Language based object systems utilize naming techniques embedded in the programming language, while distributed object systems rely on naming services provided by the distributed object system infrastructure. Typically, distributed object systems support many more objects, generally several orders of magnitude more, than language based object systems.

These and other differences create problems in a distributed object system that either do not exist or can be handled in a different way than they are in language based object systems. One of these differences is how to manage distributed object systems that support a large number of objects, i.e., how to manage its scaling problems.

Scaling affects a number of distributed object system services, including naming, lifecycle, resource management, error control, to mention a few. This panel addresses the problems of scaling with respect to a distributed object system's security.

There are several ways one might attack the scalability of security in distributed object systems. One that has received a great deal of attention recently utilizes the concept of an object domain. Simply stated, a domain is a set of objects collected together so that an administrator can manage those objects as a unit. Each domain (called in this context a *security policy domain*) is associated with a particular security policy that may control one property or service, for example:

- the authenticated identity associated with the objects,
- how the system controls access to the objects,
- how the objects' methods are securely invoked and results securely returned to a caller,
- how access rights are delegated between objects in the domain and objects in other domains, and
- how use of objects in the domain is audited.

Associating a policy with a domain can significantly reduce the level of administration necessary in a distributed object system. Instead of establishing and managing policies for each individual object, the administrator does this for groups of objects.

However, security policy domains also introduce a number of significant problems, including:

- When the policy associated with a domain is modified, the members of the domain generally do not recognize the change at the same time. There is an interval of time when the old policy controls some objects in the domain, while the new policy controls the other objects. This can introduce security hazards that are difficult to identify and eliminate.
- The concept of a security policy domain is predicated on the assumption that a single authority controls the security properties of all domain objects. This is rarely true in practice. For example, the policy that controls access to an object may have both mandatory and discretionary components. Generally, mandatory policy is set by organizations,

while discretionary policy is set by individuals. This introduces complexity in the management of the access control policy.

- There are many cases for which the security of a distributed object system is not easily expressed in terms of a centrally managed set of objects. For example, the level of message security necessary when an object method is invoked (e.g, whether confidentiality and/or integrity is required, what crypto-algorithms should be used) is a property of the pair of objects (i.e., the client object and target object) as well as of the path taken by the messages. Using security policy domains requires the interaction of two or more policies, one associated with the client, another associated with the target and, potentially, others associated with resources along the path of the messages. The composition of these policies can result in complicated and unexpected results.
- Some objects may belong to more than one security policy domain. This introduces the problem of expressing policies in a way that allows them to be composed. When domains are arranged hierarchically or overlap, the rules for composition may become complex, which can lead to less confidence that the resulting actions are correct.
- As the policy language becomes more complex to accommodate the situations described above, it becomes harder to map the constraints imposed by organizations or individuals into a policy language statement. This can lead to errors that adversely affect the distributed object system's security.
- Since the existence of a single Trusted Computing Base (TCB) for a distributed object system occurs only rarely in practice, policy decisions distributed over mutually suspicious systems are suspect. This is best illustrated by a question: who trusts these decisions and why?

The panel uses several example applications of large-scale distributed object systems to describe these problem areas and to explore how security policy domains might help. These scenarios are described below.

- **Federated Domains**-- Several large manufacturing firms wish to share product design specifications to ensure that components will assemble with appropriate tolerances upon final assembly. To share this information efficiently, each company allows direct access to these databases through specially defined object interfaces. However, each company also wants to ensure that sensitive proprietary data, such as pricing, is not available to the other manufacturers, who may be competitors. How does each company set its own security policy for access control and message protection, and yet allow controlled sharing with the other manufacturers?
- **Security Policy Updates** -- A large-scale healthcare coalition, consisting of several hospitals, outpatient facilities, and affiliated doctors' offices, shares a distributed computer-based patient records system. The overall security policy for the system, which uses role-based access control, is controlled and configured by a central administrator. The current policy allows special overrides for access control so that patient records may always be accessed in emergency conditions. Security audits reveal that this override has been abused, and patient privacy has been compromised. The administrator wishes to further restrict the override policy to apply only to after-hours when hospital staff is scarce. How does the administrator ensure that this new policy takes effect immediately over the entire system?
- **Composition of Policies** -- Military intelligence data is stored by a large number of agencies worldwide. Each agency has its own special requirements for mandatory access control, discretionary access control, and need-to-know. Agency security policies are thus a composition of these individual security policies. How do administrators customize these policies and their composition, and ensure that users honor the policies of data imported from other agencies?
- **Mutual Suspicion** -- A consumer wishes to "surf the net" using a set of distributed object services to shop for the best price and selection of some product. The customer finally selects the product from an on-line retail outlet, and wishes to directly debit his bank account and transfer the payment for the merchandise to the retailer. The problem, however, is that the customer is not familiar with the retailer and does not trust the retailer to withdraw the proper amount. Likewise, the retailer does not know the customer and won't accept credit. Can security policy domains help to confine this transaction and establish limited trust between mutually suspicious parties?